中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

武贝宁

南京大学, 南京

E-mail:

摘要 本文的写作目的主要是南京大学2021秋季学期java高级程序设计的大作业兼结课论文。其中包含大作业的开发目标、设计理念、技术细节、课程感想等。

关键词 java,面向对象, 并发控制, 网络通信, 课程感想

1 Development Goals

1.1 What game it is?

我写的游戏是一个2Drougulike射击闯关游戏，游戏涵盖了单人模式和多人模式，玩家在一个35x25的地牢房间中战斗，目标是击败所有小怪和BOSS逃出地牢。

随着时间进行会不断随机刷出怪物，当玩家将怪物全部击杀后，就可以通过传送门进入下一关。每关的地图随机生成，同时房间中会生成一些可能含有道具的罐子，子弹可以打破这些罐子，玩家拾取道具可以恢复血量或者强化自己的属性。游戏流程共有三关，前两关都是小型怪物，且怪物生成数量会有限制，第三关会出现BOSS，BOSS被击败前会无限召唤小怪，因此玩家必须在优先攻击BOSS的同时小心小怪的子弹。

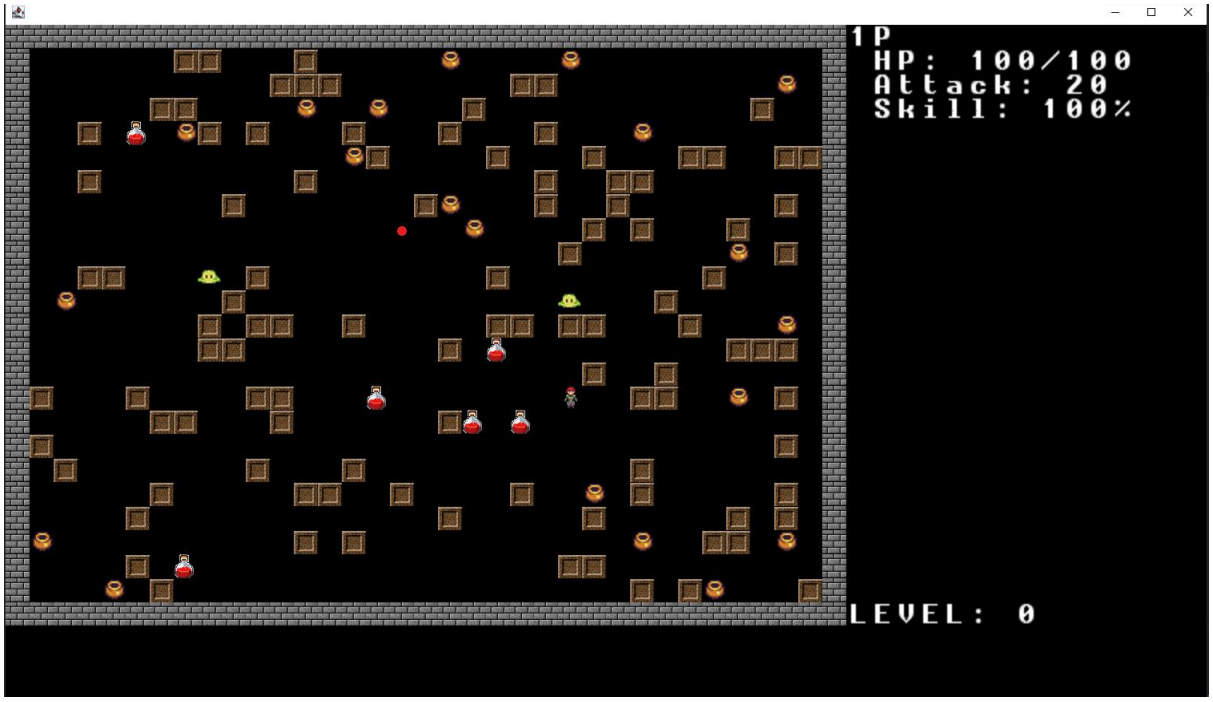

图 1　游戏界面

Figure 1　WorldScreen

1.2 Inspiration

开发最初的灵感来源是元气骑士，因为我之前沉迷过此游戏一段时间，也有着一个开发游戏的念头，因此想借着这次大作业顺便实现自己以前的一个想法。

但是实现过程中发现很多实现起来有难度的问题，例如，元气骑士中角色的射击是可以向任意角度发射，但项目的原框架采用的是2D方格式地图，每一个Tile是一个显示单位，那么想要实现任意角度发射的功能就势必需要对整个框架进行重构。鉴于课程作业的目的只是让我们掌握java知识而不是游戏开发，那么我决定将角色的攻击方式只设为向上下左右四个方向发射子弹，虽然自由度有一定削减，但是跟怪物博弈对战的乐趣还是在的。

1.3 The Game Manual

在游戏的主界面，玩家可选择单人模式(Single Game)或三人模式(Multiplayer Game)

游戏开始界面如图2所示，玩家可通过up/down键选择模式。

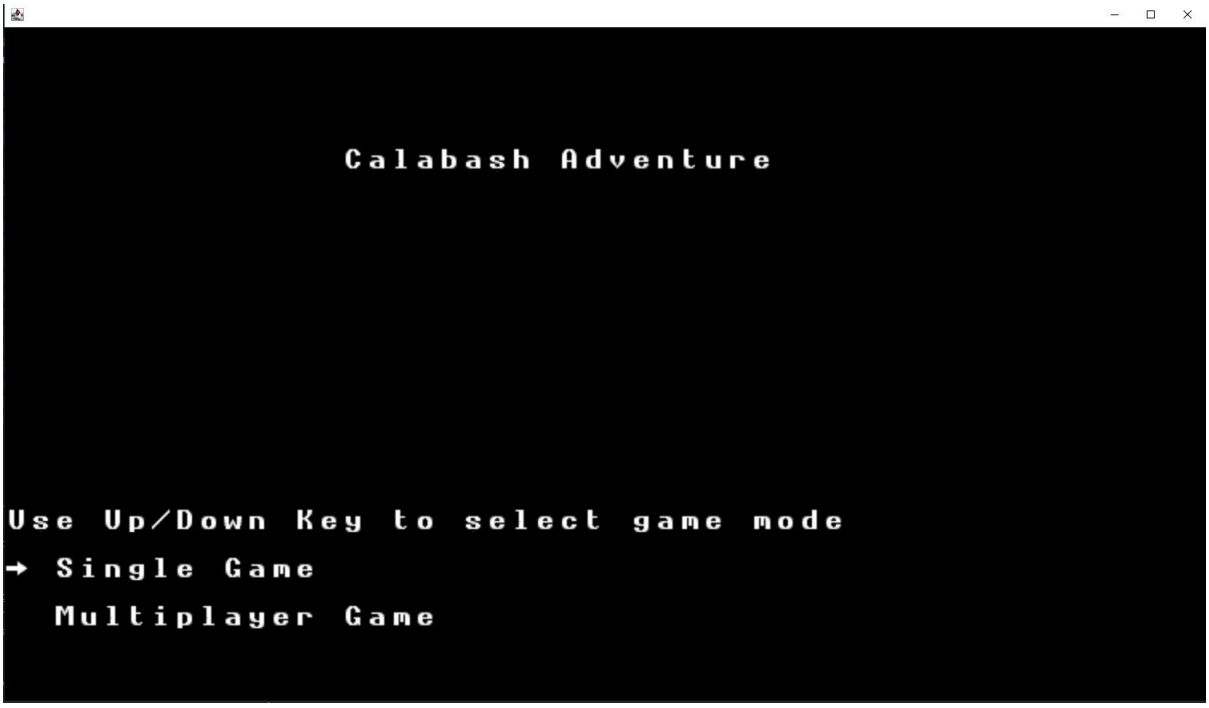

图 2　游戏开始界面

Figure 2　StartScreen

1.3.1　Operation

玩家移动：W/S/A/D

发射子弹：↑ / ↓ /←/→

使用技能：空格

多人模式下启动游戏：Enter

1.3.2　Single Game

如果本地含有上次的存档，玩家可以选择是否继续上次的游戏。

进入游戏前可以选择角色，不同角色有着不同的技能，但基础属性相同。

目前游戏共开放三个角色可供选择：大娃，二娃，三娃以下是技能介绍：

大娃：5s内攻击力提高30，cd20s

二娃：对距离自己最近的妖怪造成50点伤害，cd3s

三娃：5s内受到的伤害减少50%，如果当前血量低于30%，则恢复已损失生命值的50%，cd20s

1.3.3　Multiplayer Game

进入多人模式需要开启多个终端，其中需要一个Server和最多三个Client。

首先开启Server，然后再开启Client1,Client2,Client3(顺序不限)，当所有玩家准备就绪后，其中一位玩家按Enter键即可开始游戏。

多人模式下，默认1P是大娃，2P是二娃，3P是三娃, 当一个玩家死亡后，其屏幕跳转到Lose界面，其他玩家不受影响，可以继续战斗。(后续考虑加入复活机制)
当BOSS被击败后，存活的所有玩家跳转到胜利界面。

2 Design Idea

本项目的设计继承了jw04Rougelike分支下原本的框架结构，在此基础上增加了一些类，下面进行详细介绍

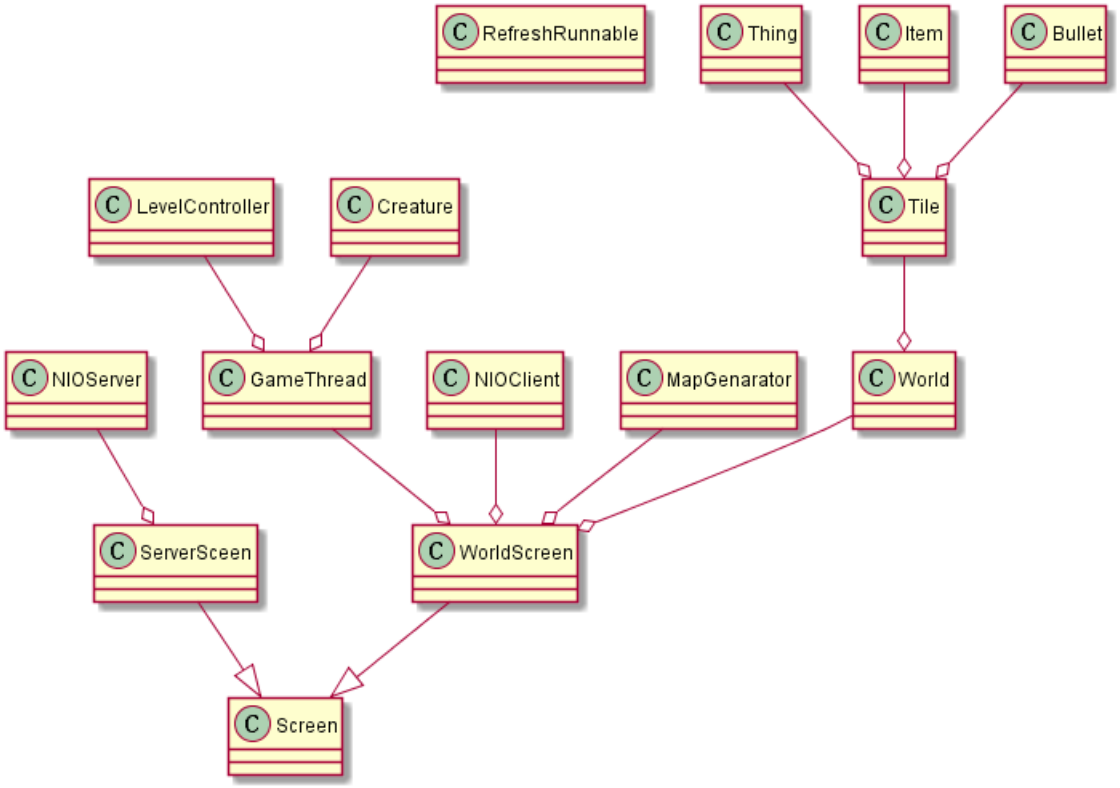

图 3　类的包含关系图
Figure 3　MultiplayerGame

2.1 Screen

负责界面显示，其中WorldScreen是游戏画面显示的主类，其他类则是负责游戏模式的选取或是游戏结束的表示。

2.1.1 WorldScreen

游戏的主体窗口，主要包含一个World对象存储所有游戏内容，通过接收用户按键使游戏状态做出改变，

2.1.2 Other Screens

StartScreen负责显示开始界面。

LoadDataScreen负责显示单人游戏下读取本地存档，以显示当前是否可以继续游戏。

SelectScreen负责单人游戏模式下选择角色。

NetworkSelectScreen负责多人游戏模式下选择Server与Client。

ServerScreen,WinScreen,LoseScreen负责输出相关游戏信息(Server运行中、游戏胜利、游戏结束)

2.2 World

World类：由35x25个tiles组成，容纳所有单位，是游戏世界的主体

2.2.1 Tile

在jw04中，每个Tile仅可容纳一个Thing，在这里为了添加物品、子弹机制，因此对Tile进行了重构。

重构后的Tile可以容纳一个Thing,一个Item,若干个Bullet.

2.2.2 Mapgenarator

地图生成类，给定一个种子进行初始化，给定一个地图系数以随机生成地图。

2.3 Thing

Thing类：所有单位的基类，在此类上衍生出Item类，Creature类

2.3.1 Item

能被玩家拾取的所有物品，目前有三种Item：Heart,HeartCystal,Sword

Heart:玩家拾取后可以恢复20点生命值

HeartCystal:玩家拾取后可以增加20点生命值上限并恢复20点生命值

Sword:玩家拾取后可以增加10点攻击力

在玩家的updateState方法中，会随时检测当前格子中是否有Item，如果有则拾取该Item并进行效果结算。

2.3.2 Creature

Calabash和Monster的父类，每个Creature都是一个runnable

Calabash类：代表玩家所操控的角色，目前共有三个子类(三种角色)，父类中实现了三种葫芦娃公有的移动和攻击方法，子类各自实现其技能方法。

Monster类：代表地牢中的怪物，目前共有三个子类(两种小怪Frog和Bat，一种Boss Snake)同样，Monster实现了三种怪物公有的移动和攻击方法，子类各自实现其具体数值和行动方式。

2.3.3 Bullet

子弹类，包含玩家发出的子弹CalabashBullet和怪物发出的子弹MonsterBullet。

2.3.4 Ohter Things

剩下的Things主要是一些地图要素，例如空地Floor，墙壁Wall，不可击穿的障碍物Barrier，可被打破的罐子Jar。

2.4 Data

游戏存档功能实现类，在类中实现了存档函数Save和读档函数Load，可以将当前的游戏状态存入本地文件，或从本地文件读取存档。

2.5 LevelController

游戏关卡控制类，本身是一个Runnable，负责关卡中怪物和药物的生成。

2.6 GameThread

游戏线程控制类，负责统筹所有生物线程以及关卡控制线程。

2.7 RefreshRunnable

用于实现屏幕刷新的一个线程，运行程序时即启动。

2.8 network

NIOServer和NIOClient类，用于网络通信，将在3.2中详细介绍。

3 Technical

3.1 Concurrent control

本游戏的并发控制主要采用了线程池模式，在游戏中，每个生物都是一个Runnable，游戏通过一个类GameThread来控制，WorldScreen类中有一个gameThread对象。

当游戏正式开始(单人模式初始化即开始，多人模式需要一位玩家按下enter)后，WorldScreen调用gameThread的start方法，启动所有生物线程和levelController线程。

生物线程的run方法主体是一个while循环，当HP¡=0时退出循环并进行相关死亡处理。在循环中，每隔一定时间(这里我使用TimeUnit下的sleep函数实现)就调用就更新一次状态。

考虑到一个生物只能占据一个Tile，因此不同线程间存在并发问题，因此我将生物的移动函数goUp, goDown,goLeft,goRight添加了synchronized,这样一个线程调用该方法的时候，其他线程就不能调用了，以避免并发产生的问题。

刚开始设计时，我想过是否要将每个子弹也作为一个线程，但是尝试过后发现这样的开销太大了，程序经常会崩溃，因此放弃了这个想法。

除了生物线程之外，还有一个线程负责屏幕的刷新，这个线程在程序启动时即启动，每隔一定时间调用Screen的Refresh的方法进行刷新。

3.2 Graphical UI

感谢老曹提供的AsciiPanel，大大简化了本游戏的图形化难度，只需要从网上找到合适的字符集替换原有的resources中的图片，然后修改AsciiPanel文件中的paint函数，将其按照颜色绘制图片的代码修改为直接显示图片即可。(同时感谢群友分享的游戏制作资源网站，大大节省了我找素材的时间)

3.3 Input and Output

本游戏只在单人模式下存档和读档时使用到了输入输出机制。

Data类是为了方便存档和读档所设计的类，其中存有单人模式下的关卡数、地图、玩家选择的角色类型、生命值、最大生命值、攻击力。这些信息按照特殊方式由load函数存储进游戏当前path下的SaveData.txt中，Data类可以通过Save方法从中读取信息。

我在这里使用的是字节流而非数据流，导致输入输出的代码可读性很差，需要频繁的读取特殊字符(当时没想到使用数据流，惭愧)

3.4 Network communication

本游戏后续加入了多人模式，采取的通信方式是NIO Selector。为了实现网络通信，设计了两个类NIOServer和NIOClient，即服务器和客户端。

其中NIOServer是ServerScreen的成员，当玩家选择服务端选项时跳转到该界面并进行NIOServer的初始化。NIOClient是WorldScreen的成员，当玩家选择客户端选项时跳转到游戏界面并准备进行连接。

Server只负责接收和广播信息，Client负责发送信息。当某个Client做出一个操作时，向Server端发送这个操作，然后Server对收到的消息进行处理，然后根据处理结果向所有Client发送消息，每个Client根据消息在自己的客户端上做出操作。

两个类的设计参考了NIO的示例代码，查阅资料得知，目前联网游戏所采用的客户端同步方式主要有两种：帧同步和状态同步。考虑到状态同步需要实现对象的序列化，操作较为复杂，因此我选择使用帧同步进行设计。

客户端之间通信的消息是经过特殊处理的，双方在收到data之后，都需要把data转换成一个字符串然后通过handleInfo函数进行分析。

因为采用的是帧同步，那么每个客户端的各种随机数就要保持一致，因此需要由客户端统一发送一个种子。当一个客户端连接进服务器时，其会向客户端发送一个”Ask/for/seed/0/”的消息，然后服务器向每个客户端发送一个当前关卡的种子，客户端依据此种子对地图、怪物AI等进行种子设置。当一个用户做出某个操作时(例如，player1做出向下的操作，那么该客户端会向服务器发送一个”Action/1/40/”,其中83是向下移动的键码)，然后客户端分析这个字符串，向所有客户端发送这个字符串，这样每个客户端都让自己的player1执行向下移动的操作。

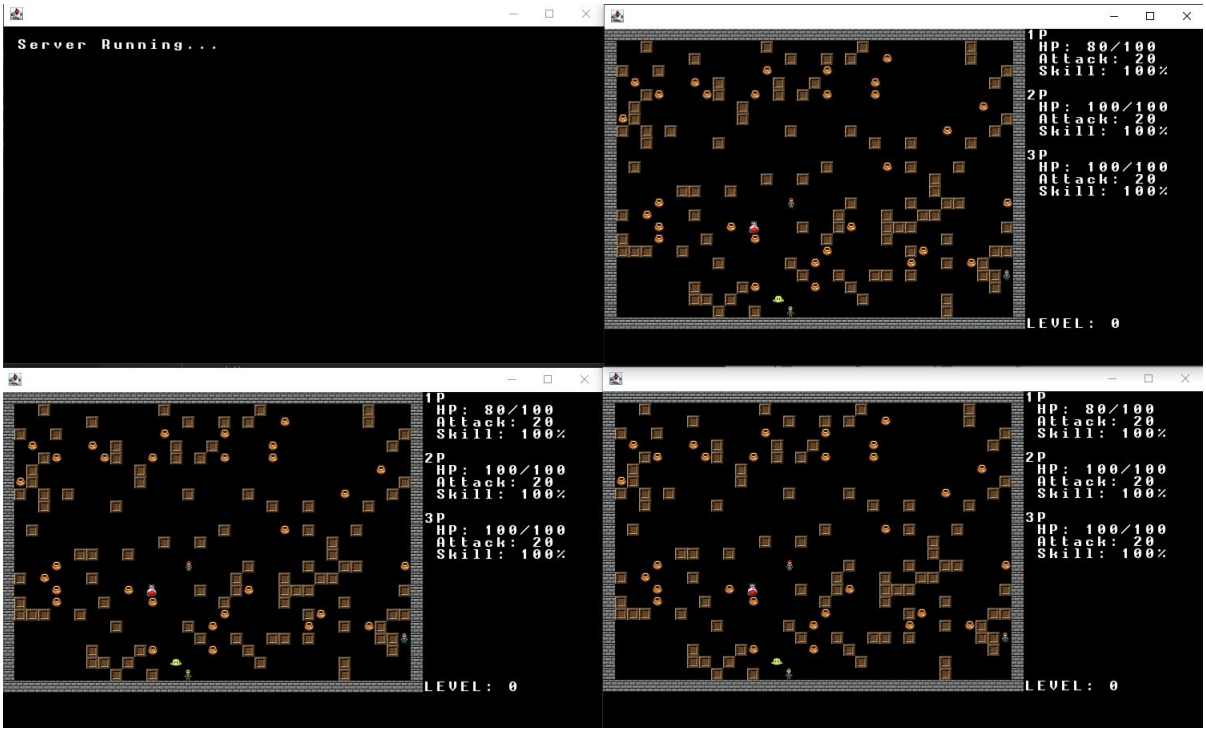

图 4　多人模式画面

Figure 4　MultiplayerGame

4　Development

4.1　Automatic construction

开发过程主要使用了MAVEN作为自动构建工具，其中添加了springframework作为构建插件，junit和jacoco作为单元测试插件。

值得注意的是，原项目是把resources文件夹放在代码同一目录，由于maven在构建时只会编译java目录下的.java文件，因此需要将resources文件夹放在main目录下，这样maven才会自动把resources文件夹复制到target中。(因此asciiPanel中的相关读取语句需要修改)

由于网络编程中使用到了java11版本才有的语句，因此java11以下的版本运行jar包无法进行多人游戏模式。

4.2　Unit Test

单元测试使用了Junit作为测试工具，测试了单人游戏模式下的大部分语句，工程总方法测试覆盖率达到63.9%。

Calabash-Adventure

Calabash-Adventure

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
screen		40%		21%	133	177	321	508	35	66	7	9
network		0%		0%	35	35	140	140	16	16	2	2
asciiPanel		44%		44%	47	63	60	135	13	25	0	2
creature		64%		43%	65	109	80	243	10	48	0	3
creature.monsters		67%		37%	45	68	64	205	2	18	0	3
level		42%		4%	20	25	34	71	0	4	0	1
bullet		66%		50%	14	27	35	97	1	12	0	3
util		69%		74%	18	52	32	111	12	36	2	12
creature.calabashes		78%		59%	13	31	20	94	0	10	0	3
threadController		62%		50%	5	9	11	29	2	6	1	2
default		55%		n/a	4	6	12	25	4	6	0	1
gamedata		97%		83%	6	29	3	95	0	11	0	1
map		100%		100%	0	18	0	31	0	5	0	1
Total	3,554 of 7,722	53%	445 of 730	39%	405	649	812	1,784	95	263	12	43

图 5　单元测试结果

Figure 5　Unit Test Result

5　Thoughts and Seggestions

在这门课上真的学到很多，老曹也是一位很负责任的老师。虽然这门课的作业有些挑战，但是也只有在亲自做作业写代码的过程中才能熟练掌握java的各种语法和机制，才能体会到面向对象的理念和项目开发的艰难。模块间耦合性强、各种并发问题，这些东西不是靠看书看例子就能纸上谈兵解决的，得亲自动手才能理解更加深刻。